Advanced Regression: 4b Machine learning: Ensemble methods

Garyfallos Konstantinoudis

Epidemiology and Biostatistics, Imperial College London

14th March 2023

Ensemble methods

- Bagging
- Random forests
- Boosting
- Variable importance
- Ensemble methods in R

Bootstrap aggregation (bagging)

- Let the original dataset contain n samples
- Take $n' < n$ repeated samples (with replacement) of the original dataset to create B new datasets ($b \in 1, ..., B$ bootstrapped training datasets).
- Grow a tree on each bootstrapped training datasets.
- Each of the trees has little bias, but high variance.

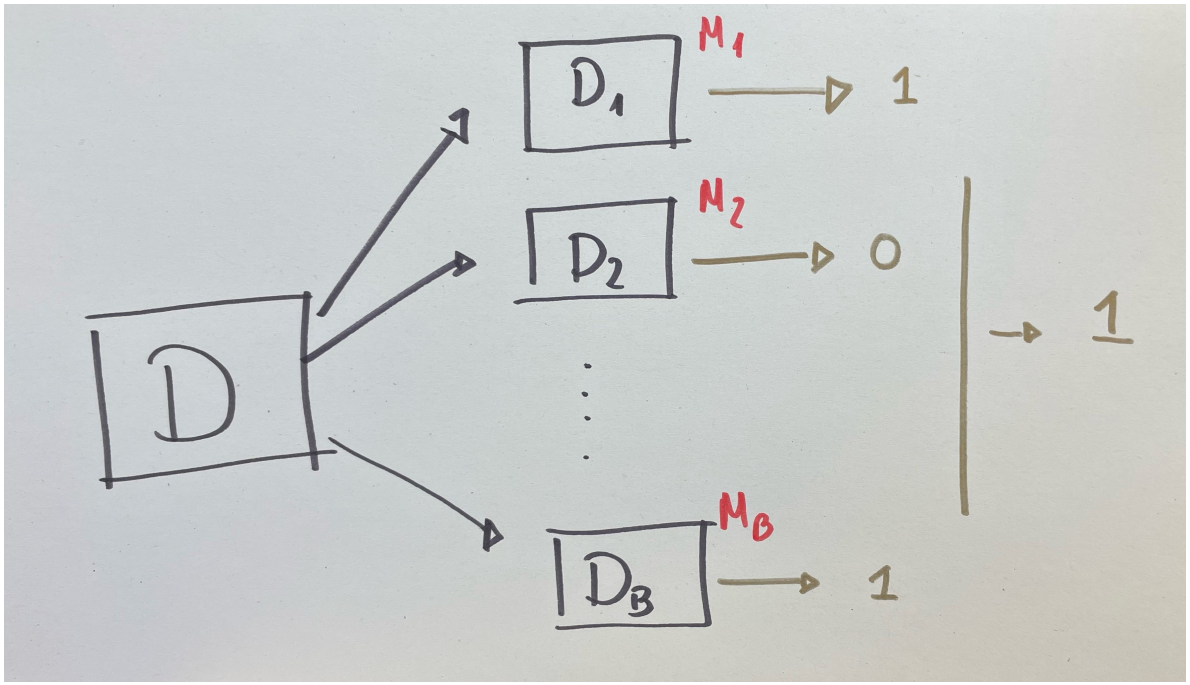
Average over trees

- Regression tree: Compute the mean over the B predictions f_b of each individual regression tree

$$f_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} f_b(x)$$

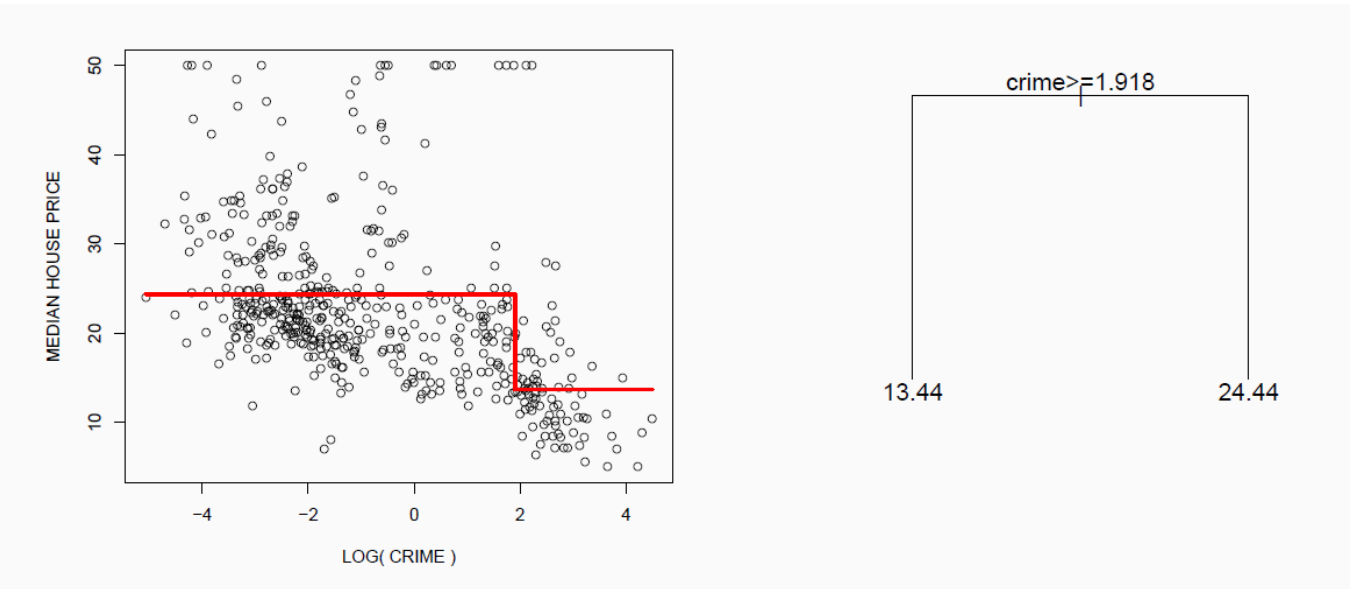
- Classification tree: Majority vote (mode), the final prediction is the one that occurred most frequently among the B trees

Bagging



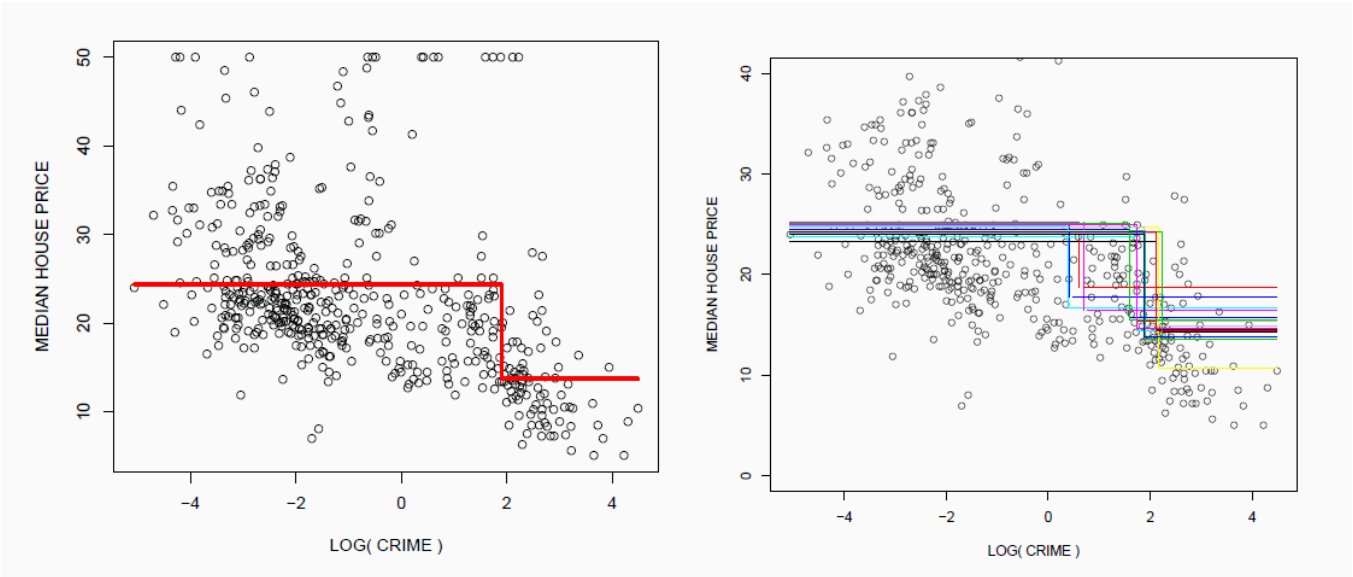
Example (taken by Prof Filippi lecture)

- Regression for Boston housing data
- Aim: predict median house prices based only on crime rate
- Consider a tree with a single split at the root



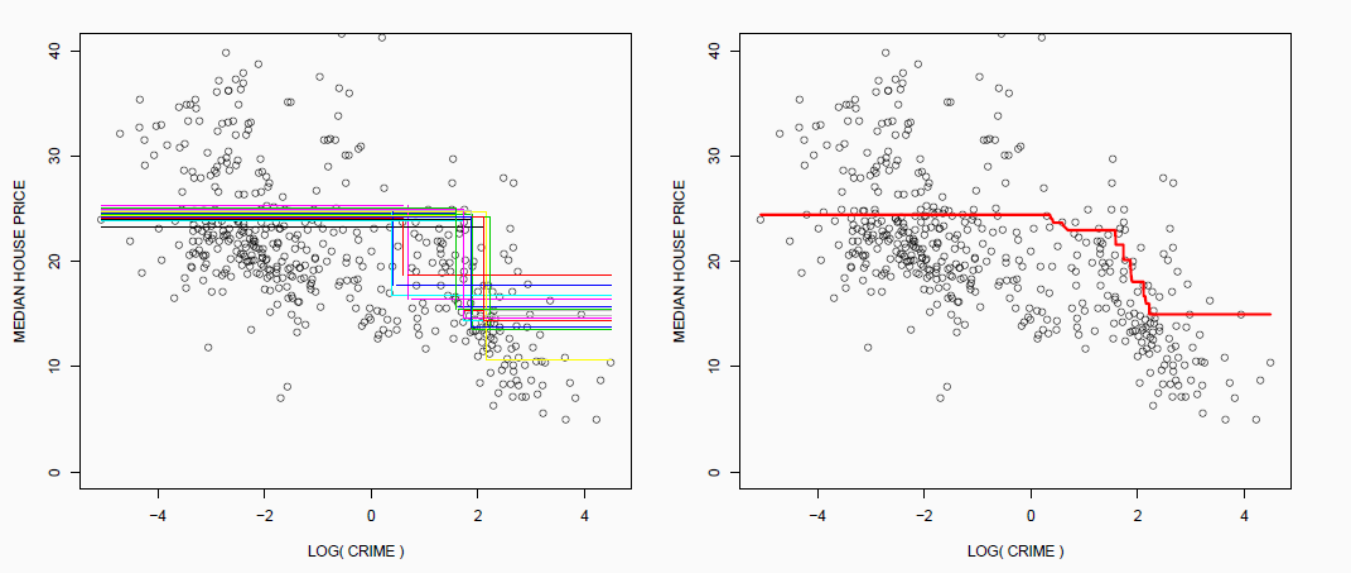
Example (taken by Prof Filippi lecture)

- 20 Bootstrap samples



Example (taken by Prof Filippi lecture)

- Summarize the samples



Out-of-bag error

- Bagging has an intrinsic way of evaluating prediction performance

Out-of-bag samples

- The bootstrapped training dataset contains n' samples drawn at random.
- $n - n'$ samples are left out. They are the out-of-bag samples.

- The out-of-bag error can be used to estimate the test error and prediction performance.
- No cross-validation is needed.

Random forest

- Bagging always considers all p variables to build a decision tree.
- Consequently, individual trees in bagging may look very similar.
- Random forest selects at random a subset of m variables to be considered at each split.
- Consequently, individual trees in random forests may look very different (or random).

How to select m?

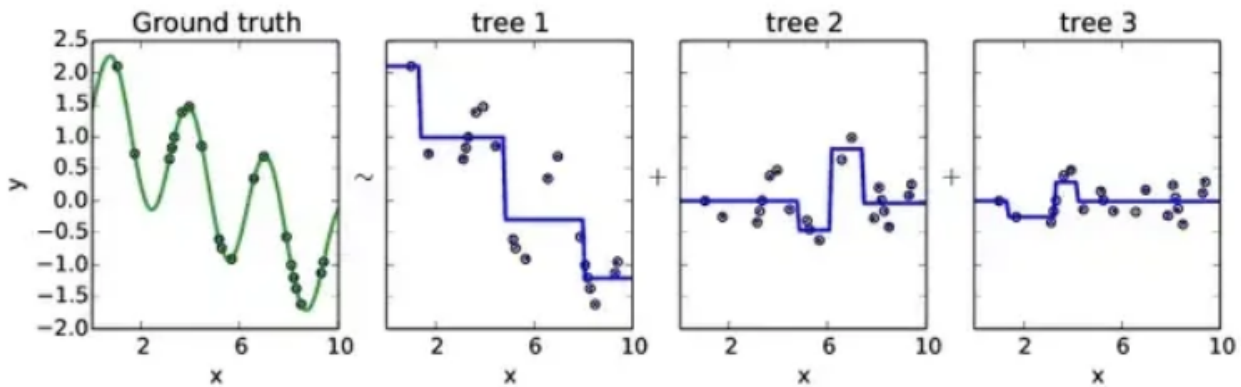
- Regression random forest: $m = p/3$
- Classification random forest: $m = \sqrt{(p)}$
- Bagging: $m = p$

Bagging

Random forest

Boosting of trees

- Trees are grown sequentially, using information from previously grown trees.
- Boosting learns from mistakes, after fitting the first tree, further trees are grown based on the residuals.
- Tuning parameters
  1. B: Number of trees
  2. λ: Learning rate
  3. d: Number of splits per tree, interaction depth



Boosting of trees

Boosting algorithm:

1. Initialise the output parameters: the predicted values $f(x) = 0$ and the residuals $r_i = y_i$ for all observations i.
2. Loop through $b \in 1, ..., B$, repeat:
   (a) Fit tree b with d splits, obtain new predicted values $f_b(x)$
   (b) Update
   $$f(x) = f(x) + \lambda f_b(x)$$
   (c)
   $$r_i = r_i - \lambda f_b(x)$$
3. Output as the final model the boosted predictions

$$f_{boost}(x) = \sum_{b=1}^{B} \lambda f_b(x)$$

Outlook: Gradient and XGboost

- Gradient Boosting: A special case of boosting where errors are minimized by gradient descent algorithm.
- eXtreme Gradient Boosting (XGBoost):
  - Boosting algorithm, sequential learning
  - Approximation and regularization
  - Computationally efficient, based on parallel and distributed computing

Variable importance: Understanding how algorithms work

Interpretable machine learning

- Variable importance in ensemble trees allows to rank variables by their importance in the model.

There are two approaches for variable importance:

1. **Permutation: Mean decrease in accuracy**
   - First fit the model on the original data and evaluate the prediction error on the out-of-bag samples.
   - Permute variable j, refit the model and re-evaluate the prediction error on the out-of-bag samples.
   - The mean decrease of accuracy after permuting variable j can be used to measure the importance of the jth variable.
2. **Gini-index: Mean decrease in impurity**
   - Record the decrease of impurity every time a tree is split at variable j.
   - Average over all trees.

Bagging and random forests in `library(randomForest)`

```
randomForest(x, y, xtest=NULL, ytest=NULL,
ntree=500,
mtry=if (!is.null(y) && !is.factor(y))
max(floor(ncol(x)/3), 1)
else floor(sqrt(ncol(x))),
replace=TRUE, importance=TRUE)
```

- `x,y`: training data
- `xtest,ytest`: test data (optional, usually not available)
- `ntree`: number of trees to grow
- `mtry`: number of variables to consider in each tree
- `replace`: sample with replacement
- `importance=TRUE`: computes variable importance also on out-of-bag samples

Bagging and random forests in `library(randomForest)`

`mtry` number of variables to consider in each tree

- Regression random forest: $m = p/3$
- Classification random forest: $m = \sqrt{(p)}$
- Bagging: $m = p$

```
> rf.out = randomForest(x,y)
> rf.out

Call:
 randomForest(x = x, y = y)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3

          Mean of squared residuals: 0.1800549
                    % Var explained: 42.17
>
> boost.out = randomForest(x,y,mtry=ncol(x))
> boost.out

Call:
 randomForest(x = x, y = y, mtry = ncol(x))
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 10

          Mean of squared residuals: 0.1896062
                    % Var explained: 39.1
>
```

Bagging and random forests in `library(randomForest)`

Values:

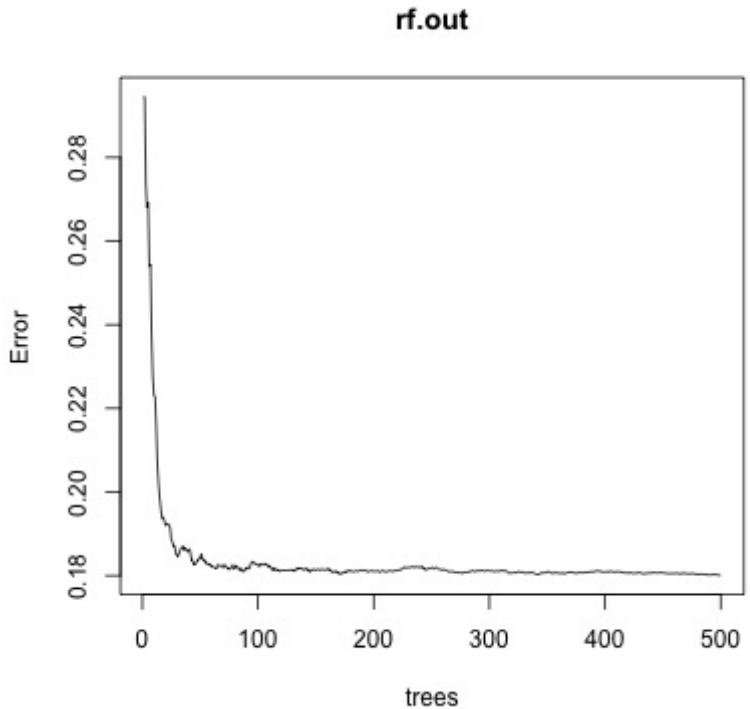
- `$predicted`: Predicted values
- Classification measures: `$err.rate, $confusion, $votes`
- Regression measures: `$mse, $rsq`
- `$importance`: Variable importance (mean decrease in accuracy and mean decrease in impurity)

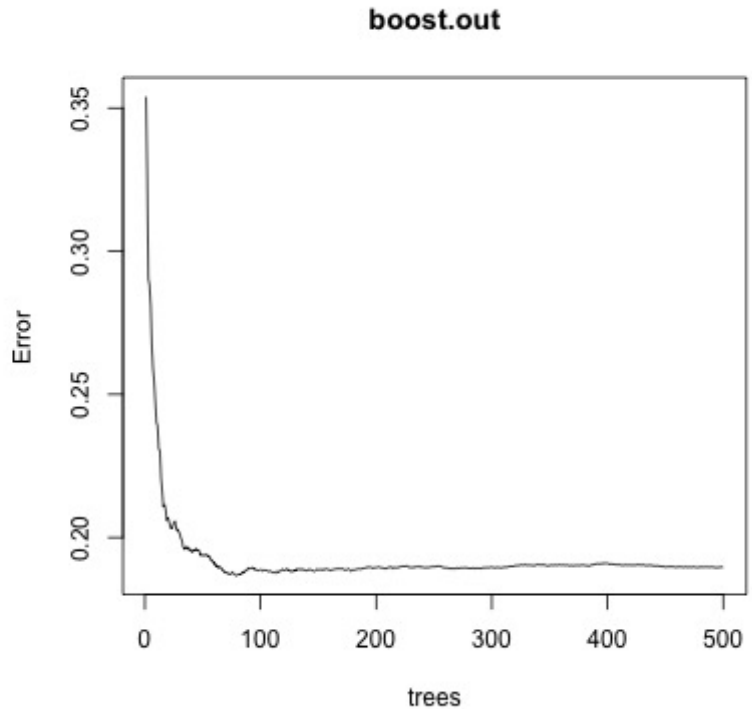
Further functions

- `predict.randomForest`: Predict new data
- `plot.randomForest`: Plot error against number of trees
- `varImpPlot`: Plot variable importance

Bagging and random forests in `library(randomForest)`

`plot.randomForest`

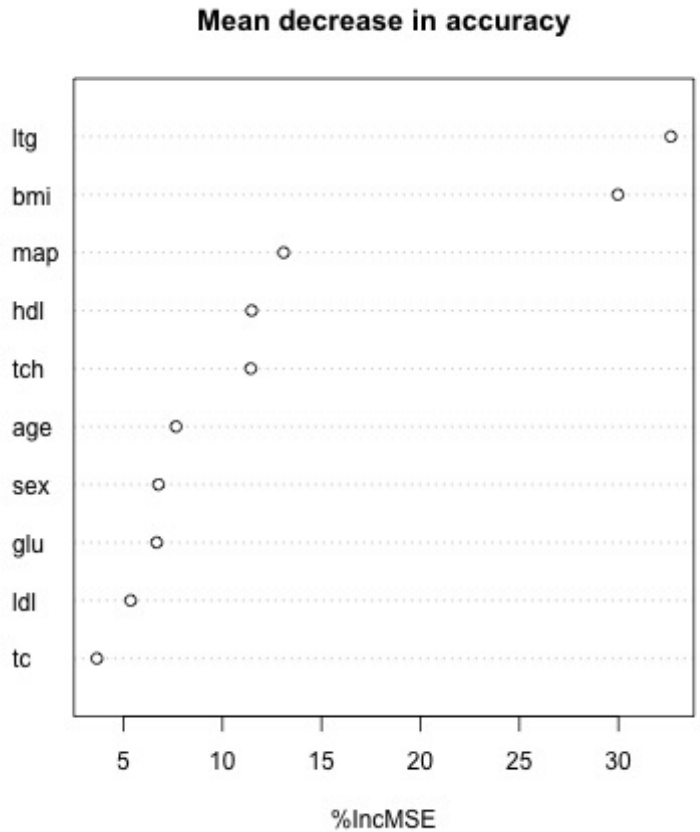


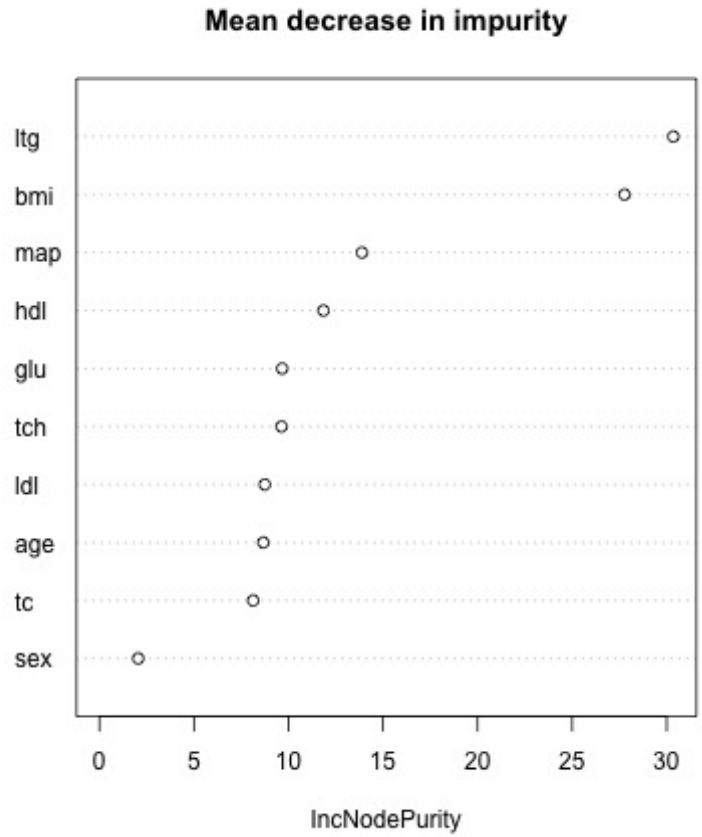


Bagging and random forests in `library(randomForest)`

`varImpPlot(rf.out, type)`

- `type=1`: Mean decrease in accuracy
- `type=2`: Mean decrease in impurity





Boosting in gbm

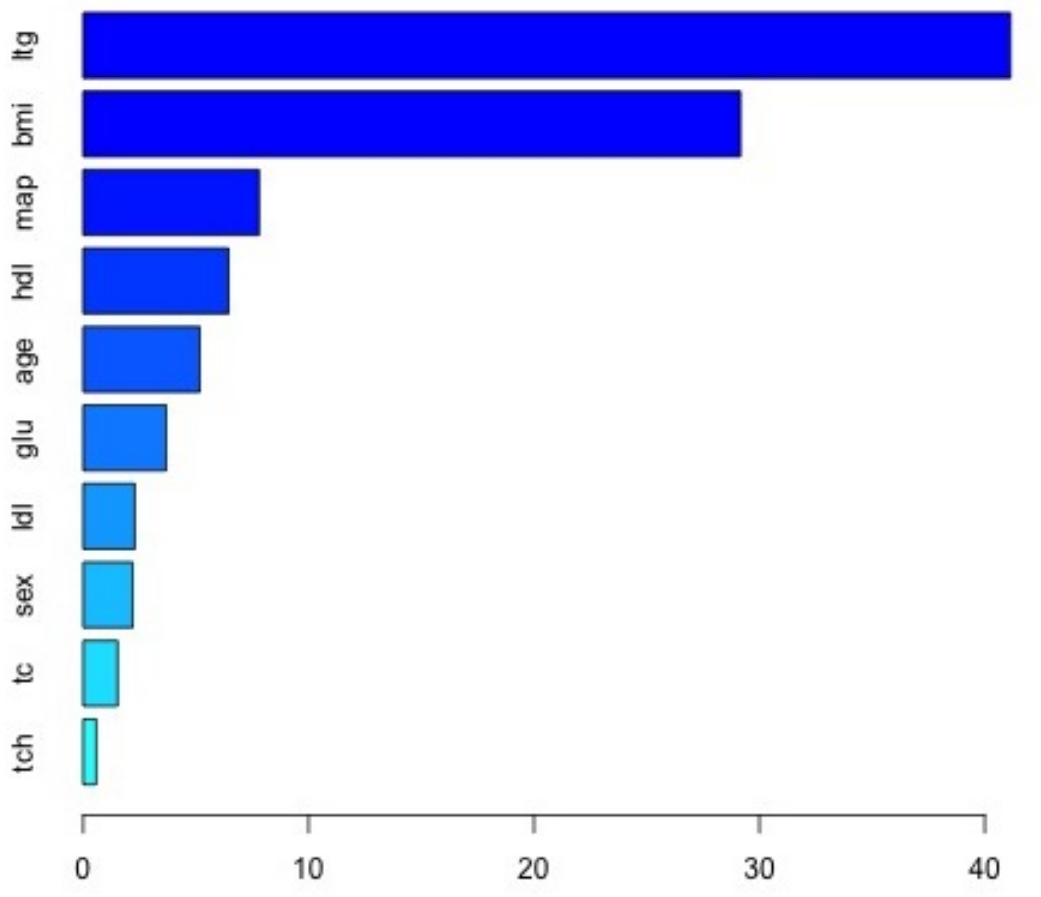
```
gbm(formula, distribution, data,
n.trees = 100, interaction.depth = 1, n.minobsinnode
= 10,
shrinkage = 0.1, cv.folds = 0)
```

- `formula`: $y \sim .$
- `distribution`: family of the outcome
- `data`: data.frame
- `n.trees`: number of trees to fit
- `interaction.depth`: maximum depth of each tree
- `n.minobsinnode`: minimum number of observations in the terminal nodes
- `shrinkage`: shrinkage factor
- `cv.folds > 0`: performs additionally cross-validation

Boosting in mboost

Further functions

- `predict.gbm`: Predict new data
- `summary`: Relative influence



Take away: Ensemble methods

- Decision trees offer great interpretability.
- But they tend to overfit, performing poorly in prediction.
- Ensemble methods like bagging, random forest and boosting, fit many trees in different variations and average over them.
- This reduces the variance and leads to greater generalizability.
- Variable importance measures help to understand the contribution of specific variables.

Thank you Deborah Schneider-Luftman for material and graphs

General background reading

An Introduction to Statistical Learning

with Applications in R

Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani

Home

About this Book

R Code for Labs

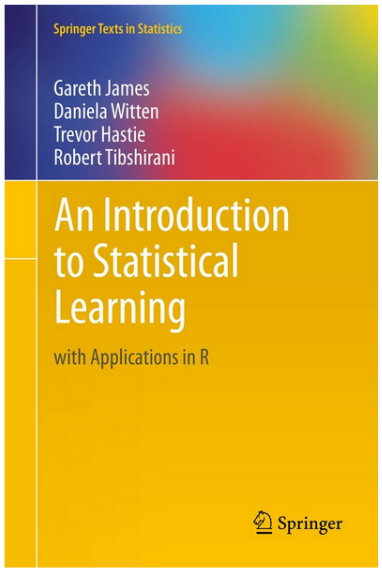
Data Sets and Figures

ISLR Package

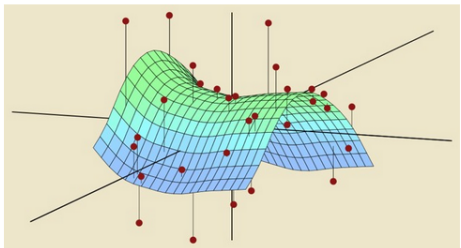
Get the Book

Author Bios

Errata



Download the book PDF
(corrected 7th printing)



Statistical Learning MOOC covering the entire ISL book offered by Trevor Hastie and Rob Tibshirani. Start anytime in self-paced mode.

- An Introduction to Statistical Learning: 8 Tree-Based Methods
- `http://faculty.marshall.usc.edu/gareth-james/ISL/`